RIDGE ASSEMBLER REFERENCE MANUAL

(RASM)

August 30,1982

The Ridge assembler (RASM) accepts source lines of Ridge instructions, pseudo instructions and assembler directives, and produces either executable object code or object code suitable for input by the linker. Following is a list of rules for the assembler source syntax.

Syntax Notation

Register numbers are indicated by (R1) or (R2). Registers are specified by Rn where "n" can be 0 - 15. Usage of "R2" indicates a literal value from 0 -15, rather than the register number. Items enclosed by braces, "{", and "}" indicate one token must be selected. Items enclosed by brackets, "[", and "]" are optional.

1. Source line input is free-form, with the restriction that the first character of a line is reserved for labels. Labels are either jump targets, pseudo instructions or assembler directives. Instructions may begin following a jump target label, or after one or more leading blanks on a line.

2. Jump targets must be followed by a ":".

3. Blank lines are ignored.

4. Input following a ";" is ignored.

5. Expressions used may contain decimal numbers, hex numbers or label names. "+", "-", "*", "/", and "(...)" may be used in expressions.

6. Hex numbers must begin with the digits 0 - 9 and must end with "H".

7. Labels must start with A - Z and may contain digits 0 - 9. Labels may be an arbitrary length, but only the first 16 characters are used to uniquely identify a label.

8. The assembler accepts both upper and lower case input, but does not distinguish between tokens that are of different case.

Assembler Directives

The assembler directives are listed below (all directives must start as the first character on a line):

LIST	Assembler displays each source line as it is read. This is the assembler default.
NOLIST	Assembler does not print source lines. Errors are still displayed, however.
HEXOUT	Assembler output is an object file suitable for linking. EXTERNAL and GLOBAL names are placed in the output file.

ALIGN {2}
{4}
{8}
	The next instruction assembled is placed on a 2, 4, or 8-byte boundary, as specified.

PAGE	Places a form feed in the output file.

Pseudo Instructions

label CODE expression	This equates a label with an expression that can be used in instructions that reference the code segment. "label" must be the first character of a line.
label DATA expression	This equates a label with an expression that can be used in instructions that reference the data segment. "label" must be the first character of a line.
ORIGIN n	The next instruction assembled is placed at byte "n" in the object code. The assembler default places the first instruction assembled in the first byte of the object file.
BLOCK count,byte	"byte" is an expression that is placed in the object code. "count" is a replication factor that must be a positive integer.
EXTERNAL name	This places the label "name" in the HEXOUT object file. All references to "name" can be resolved by the linker program. EXTERNAL must be used in the source text before any occurrences of "name".
GLOBAL name	This places the label "name" in the HEXOUT object file. This permits labels used in the assembled code module to be bound by the linker to EXTERNAL references in other code modules.

Instruction Syntax

The syntax for Ridge instructions is listed below.

Zero Register Format Instructions

```
    {FLUSH   }
    {TRAPEXIT}
    {RUM     }
```

One Register Format Instructions

```
    {ELOGR}
    {ELOGW}    (R1)
    {ITEST}
```

Two Register Format Instructions

```
     instr     (R1) , (R2)
```

Where "instr" is one of the following (as they appear in the opcode chart):

NEG
ADD
SUB
MPY
DIV
REM
NOT
OR
XOR
AND
CBIT
TBIT
SBIT
CHK

NOP

FIXT
FIXR
RNEG
RADD
RSUB
RMPY
RDIV
MAKERD
LCOMP
FLOAT
RCOMP
EADD
ESUB
EMPY
EDIV

DFIXT
DFIXR
DRNEG
DRADD
DRSUB
DRMPY
DRDIV
MAKEDR
DCOMP
DFLOAT
DRCOMP

SUS
LUS
LDREGS
TRANS
DIRT
READ
WRITE

CALLR
RET

LSL
LSR
ASL
ASR
DLSL
DLSR
CSL

Special Register Formats

```
     MOVE     { (R1)  ,  (R2)   }
              { (SR1) ,  (R2)   }
              { (R1)  ,  (SR2)  }


     KCALL    n            where "n" is an expression that results
                              in an integer from 0 -255


     TRAP     n            where "n" is an expression that results
                              in an integer from 0 - 15
```

TEST Register Format

```
     TEST     (R1) lop {(R2)}      where lop is :  >, <, =, <=,
                       { R2 }                      >=, <>
```

Branch Format

Unconditional branch:

```
     BR       target [,L]          where L indicates long (32-bit)
                                      displacement

     target is an expression
```

Call:

```
     CALL     (R1) , target [,L]   where L indicates long (32-bit)
                                      displacement

     target is an expression
```

Conditional branches:

```
     {BR  }  (R1)   lop   { (R2) }  , target[] [,L]
     {LOOP}               {  R2  }
```

where

```
     lop is one of: >, <, =, <=, >=, <>

     target (no space following target name) set branch
        prediction bit in target displacement

     L indicates long (32-bit) displacement
```

Memory Reference Format

```
{LOADB   }
{LOADH   }
{LOADHS  }
{LOAD    }
{LOADD   }
{LADDR   }    (R1) {  ,  (R2)    [, address [, L] ]}
{        }         {[,  (R2)]     , address [, L] ]}
{STOREB  }
{STOREH  }
{STORE   }
{STORED  }
{
{LOADBP  }
{LOADHP  }
{LOADHSP}
{LOADP
{LOADDP  }
{LADDRP  }
```

where
"address" is an expression

L indicates long (32-bit displacement)

Load instructions followed by "P" reference the code segment; those without reference the data segment.